DER BUBBLESORT ALGORITHMUS

Link zum GitHub-Repository
https://github.com/F-P-1611/sorting.git

INHALT

- ALLGEMEINES ZUM BUBBLESORT ALGORITHMUS
- BEST-, WORST- UND AVERAGE CASE
- VOR- UND NACHTEILE VON BUBBLESORT
- IMPLEMENTATION IN EIN JAVA PROGRAMM + BEISPIEL

ALLGEMEINES ZUM BUBBLESORT ALGORITHMUS

- STABILER SORTIERALGORITHMUS, DER ZWEI BENACHBARTE WERTE VERGLEICHT

  **STABIL BEDEUTET, DASS ZWEI GLEICHE WERTE IN DER GLEICHEN REIHENFOLGE AUSGEGEBEN WERDEN, WIE SIE EINGEGEBEN WURDEN, WENN SIE DEM SORTIERKRITERIUM GEGENÜBER GLEICH SIND - WIKIPEADIAARTIKEL DAZU*

- JE NACH SORTIERKRITERIUM WIRD NACH DEM VERGLEICH GGF. GETAUSCHT
- DIE NÄCHSTEN BENACHBARTEN WERTE WERDEN VERGLICHEN

  ⟶ DADURCH RÜCKT DER GRÖSSTE WERT AN SEINE POSITION, *WIE EINE BLASE IM WASSER NACH UND NACH AUFSTEIGT – DAHER DER NAME UND DAS PPP-TEMPLATE*

HIER KLICKEN FÜR VISUALISIERUNG BEI HACKEREARTH.COM

ALLGEMEINES ZUM BUBBLESORT ALGORITHMUS

- IN DER FOLGENDEN BUBBLEPHASE WIRD DAS ARRAY UM 1 KLEINER, DA DER WERT AUS DER LETZTEN BUBBLEPHASE, DER SICH NUN AN DER RICHTIGEN POSITION BEFINDET, NICHT MEHR BEACHTET *(S. VISUALISIERUNG HACKEREARTH.COM)*

  **EINE BUBBLEPHASE MEINT EINEN DURCHLAUF, IN DEM ALLE BENACHBARTEN ELEMENTE DES ARRAYS EINMAL VERGLICHEN WURDEN UND DER NÄCHSTGRÖSSTE WERT AN SEINE POSITION AUFGERÜCKT IST — DAS ARRAY IST NACH EINER BESTIMMTEN ANZAHL AN BUBBLEPHASEN SORTIERT*

  ⟶ *DIE ANZAHL DER BUBBLEPHASEN HÄNGT VOM ARRAY AB*

BEST-, WORST- UND AVERAGE-CASE

- **BEST CASE:** DAS ARRAY LIEGT BEREITS SORTIERT VOR

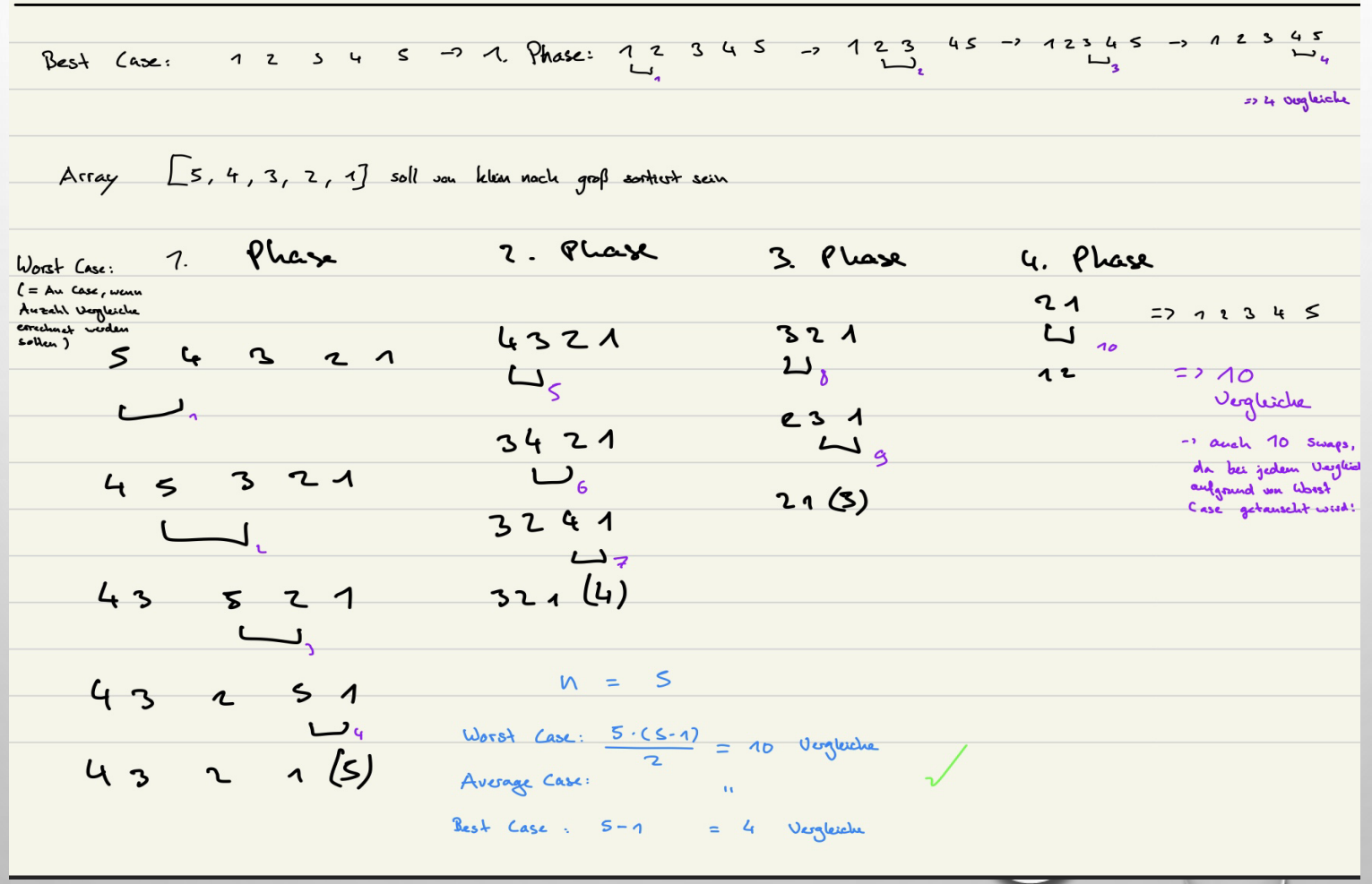
  ⟶ *EINE BUBBLEPHASE IN DER NUR VERGLICHEN WIRD, GIBT ES TROTZDEM*

  - *DIE ANZAHL DER VERGLEICHE WIRD DAHER DURCH N-1 BESCHRIEBEN, WOBEI N DIE LÄNGE DES ARRAYS BESCHREIBT*

- **WORST CASE:** DAS ARRAY LIEGT GENAU DEM SORTIERKRITERIUM ENTGEGEN GEORDNET VOR
  - ANZAHL VERGLEICHE WIRD BESCHRIEBEN DURCH (N*(N-1))/2

- **AVERAGE CASE:** DURCHSCHNITTLICHE ANORDNUNG EINES UNSORTIERT GENERIERTEN ARRAYS
  - ANZAHL VERGLEICHE WIRD BESCHRIEBEN DURCH (N*(N-1))/2

BEST-, WORST- UND AVERAGE-CASE



VOR- UND NACHTEILE

VORTEILE

- LEICHT ZU ERKLÄREN UND IN CODE ZU IMPLEMENTIEREN
- EINFACH DARZUSTELLEN
- ZUVERLÄSSIG ZUM ÜBERPFRÜFEN VON BEREITS SORTIERTEN STRUKTUREN

NACHTEILE

- ZEITKOMPLEXITÄT (INEFFIZIENT) – LAUFZEIT: $O(N^2)$ ⟶ VIELE SCHRITTE

⟶ ANDERE SORTIERALGORITHMEN WEDEN MEIST VORGEZOGEN, DA SIE SCHNELLER UND EFFIZIENTER SIND HTTPS://WWW.YOUTUBE.COM/WATCH?V=JMID4DQ6P44

IMPLEMENTATION IN EIN JAVA PROGRAMM

```
214
215     public void bSort_ActionPerformed(ActionEvent evt) {
216         // Sortieren der Zahlen im array unsorted aufsteigend
217
218         // Deklarieren der lokalen Variablen
219         int tempVar = 0;
220         int laenge = nfLaenge.getIntValue();
221         int anzahlVergleiche = 0;
222         int anzahlVerschiebungen = 0;
223
224         // Beginn äußere for-Schleife
225         for (int i = 0; i < laenge - 1; i++) {
226             // Beginn innere for-Schleife
227             for (int j = 0; j < laenge - i - 1; j++) {
228                 anzahlVergleiche++; // Anzahl Vergleiche um 1 erhöhen
229                 // Vergleich der benachbarten Werte
230                 if (unsorted[j] > unsorted[j + 1]) {
231                     anzahlVerschiebungen++;          // Anzahl Positionswechsel bzw. Tausche bzw. Verschiebungen um 1 erhöhen
232                     tempVar = unsorted[j + 1];      // Wert in temporäre Variable verstauen
233                     unsorted[j + 1] = unsorted[j];  // größeren Wert in höhere Array-Position einsetzen
234                     unsorted[j] = tempVar;          // kleineren Wert in niedrigerere Array-Position einsetzen
235                     // Tausch abgeschlossen...
236                 }
237             } // Ende innere for-Schleife
238         } // Ende äußere for-Schleife
239
240         // Aufrufen der printResult-Methode
241         printResult();
242
243         // Aktualisieren der GUI-Elemente
244         nfAnzahlPositionswechsel.setText(Integer.toString(anzahlVerschiebungen));
245         nfAnzahlVergleiche.setText(Integer.toString(anzahlVergleiche));
246
247     } // end of bSort_ActionPerformed
248
249     // Ende Methoden
250     // end methods
251 } // end of class MoD_sort
```

- LINK ZUM SOURCECODE IM REPO: HTTPS://GITHUB.COM/F-P-1611/SORTING/BLOB/MAIN/SRC/MOD_SORT.JAVA
- LINK ZUR DOKUMENTATION: HTTPS://GITHUB.COM/F-P-1611/SORTING/BLOB/MAIN/INFORMATIK-KLAUSURERSATZ-DOKUMENTATION.PDF

IMPLEMENTATION IN EIN JAVA PROGRAMM

- VERSCHACHTELN VON FOR-SCHLEIFEN:
  - DIE HAUPTSCHLEIFE BESTIMMT DIE LÄNGE DES ARRAYS (DA SCHLIESSLICH NACH JEDER BUBBLEPHASE DAS ARRAY UM 1 VERKÜRZT WIRD, DA DER GRÖSSTE WERT ZUVOR AN SEINE POSITION GERÜCKT IST)
  - DIE NEBENSCHLEIFE SORGT FÜR DAS AUFRÜCKEN "DES ZEIGERS"

    DER "ZEIGER" IST IN DIESEM FALL DIE VARIABLE, DIE DIE INDEXSTELLEN BEINHALTET (J UND J+1)
- DAS IF-STATEMENT PRÜFT DIE BEIDEN BENACHBARTEN WERTE AUF DAS SORTIERKRITERIUM
  - IM FALL, DASS DAS SORTIERKRITERIUM VERLETZT WIRD, TAUSCH EINLEITEN:

(NÄCHSTE FOLIE: TAUSCHBEISPIEL) – SIEHE AUCH IN DOKUMENTATION

IMPLEMENTATION IN EIN JAVA PROGRAMM - TAUSCHBEISPIEL

[3, 1, 4, ...] (SOLL VON KLEIN NACH GROß GEORDNET SEIN)

^ ^

VARIABLE J K

(J) = 3; (K) = 1; –> 1 IN TEMPVAR –> 3 IN (K) –> (J) UND (K) HABEN WERT 3,

WORAUS FOLGT: [3, 3, 4, ...]

–> TEMPVAR IN (J)

[1, 3, 4, ...]

^ ^

VARIABLE J K –> J = 1; K = 3; –> [1, 3, 4, ...] – TAUSCH ABGESCHLOSSEN

IMPLEMENTATION IN EIN JAVA PROGRAMM

- UM ANZAHL DER VERGLEICHE AUSGEBEN ZU LASSEN, ZÄHLERVARIABLE DEKLARIEREN UND BEI JEDEM DURCHLAUF DER NEBENSCHLEIFE UM 1 ERHÖHEN
- UM ANZAHL DER VERSCHIEBUNGEN / SWAPS AUSGEBEN ZU LASSEN, ZÄHLERVARIABLE DEKLARIEREN UND BEI JEDEM DURCHLAUF DES TAUSCHES UM 1 ERHÖHEN

BITTE DOKUMENTATION UND CODE BEACHTEN!